NOTE: This draft has not been approved and is subject to modification.
DO NOT USE FOR ACQUISITION PURPOSES.

Security Supplement

to the

Software Communications Architecture Specification

MSRC-5000SEC
V1.0
May 17, 2000

Prepared for the
Joint Tactical Radio System (JTRS) Joint Program Office

Prepared by the
Modular Software-programmable Radio Consortium
under Contract No. DAAB15-00-3-0001

Revision Summary

1.0	release for prototype implementation and validation

Changes from the previous revision, other than editorial corrections, are marked with change bars in the margins.

Table of Contents

1 INTRODUCTION 1-1
1.1 Scope. 1-1

2 OVERVIEW 2-1
2.1 Security Services. 2-1
2.2 Architectural Overview. 2-4

3 SECURITY CLASSES AND SERVICES. 3-1
3.1 CF Security Functions. 3-1
3.1.1 View INFOSEC Resources. 3-1
3.1.2 Erase Algorithm. 3-1
3.1.3 Load Internal Algorithm. 3-1
3.1.4 Load External Algorithm. 3-1
3.1.5 Support External INFOSEC. 3-2
3.1.6 Support Control, Data, and Protocol ByPass. 3-2
3.1.7 Support Multi-Channel Operation. 3-2
3.1.8 Support Declassified System. 3-2
3.1.9 Control Access. 3-2
3.1.10 Support Firewall. 3-2
3.1.11 Authenticate Users. 3-2
3.1.12 Identify Resources. 3-2
3.1.13 Control Data Path. 3-2
3.1.14 Monitor Security Policy. 3-3
3.1.15 Monitor Security Status. 3-3
3.1.16 Audit Events. 3-3
3.1.17 Support Data Separation. 3-3
3.1.18 Support Process Separation. 3-3
3.1.19 Support Classified Applications. 3-3
3.1.20 Support Classified Algorithm Storage. 3-3
3.1.21 Maintain Internal and External Data Integrity. 3-4
3.1.22 Storage Commands (HCI). 3-4
3.1.23 Support Memory Separation. 3-4
3.1.24 Establish Data Paths. 3-4
3.2 Security Bypass. 3-4
3.2.1 Introduction. 3-4
3.2.2 Types of Bypass. 3-5
3.2.3 Services Required. 3-6
3.3 General Security Requirements. 3-6
3.3.1 Security and Alternate Networking Transfer Mechanisms. 3-6
3.4 Security APIs 3-6

4 COMMON SERVICES AND DEPLOYMENT CONSIDERATIONS 4-1
4.1 Common System Services. 4-1

4.2 Operational and Deployment Considerations. 4-1

5 ARCHITECTURE COMPLIANCE 5-1
5.1 Certification Authority. 5-1
5.2 Responsibility for Compliance Evaluation. 5-1
5.3 Evaluating Compliance. 5-1

List of Figures

Figure 2-1. Security Functional Decomposition 2-1
Figure 2-2. Examples of Security Devices and Resources 2-4
Figure 2-3. ApplyLabel Use Case 2-5
Figure 2-4. Processing of Data from the Guard Bus 2-6
Figure 2-5. Example of Creation of a Port–Specific Guard 2-7
Figure 2-6. Example Security Device Guard Creation 2-8
Figure 3-1. INFOSEC Bypass 3-5

List of Tables

Table 2-1. Allocation of Security Functional Requirements to SCA Elements 2-2

1 INTRODUCTION

This Security Supplement to the Software Communications Architecture (SCA) specification establishes general implementation requirements for the development of Joint Tactical Radio System (JTRS) SCA-compliant radios. These requirements are comprised of interface specifications in the form of application program interface (API) Service Definitions and security requirements. The goal of this supplement is to provide for portability and reusability of Security related software and hardware between JTRS products and to ensure interoperability of products developed for JTRS using the SCA.

1.1 SCOPE.

The requirements in this document are derived from military security and radio system requirements and originate in the need to maintain secure communications and protect the signals-in-space as well as the data flowing through a JTRS radio itself. Security requirements are described in terms of required security functions which in turn are translated into security services required of the elements of the SCA: Core Framework, CORBA Middleware, Operating Systems, Hardware Classes, and Software Applications.

The conventions and terminology defined in the SCA specification apply to this supplement.

2 OVERVIEW

2.1 SECURITY SERVICES.

JTRS Security requirements are enabled by elements of the SCA and by applications providing security services (security functions). The security services required of a secure, programmable communications system are shown in figure 2-1, grouped into major service categories of Encrypt/Decrypt, Manage Key & Algorithm, Support Integrity & Authentication, Grant Access & Monitor Security Operations, Separate Processes & Data, and Establish Security Policy.

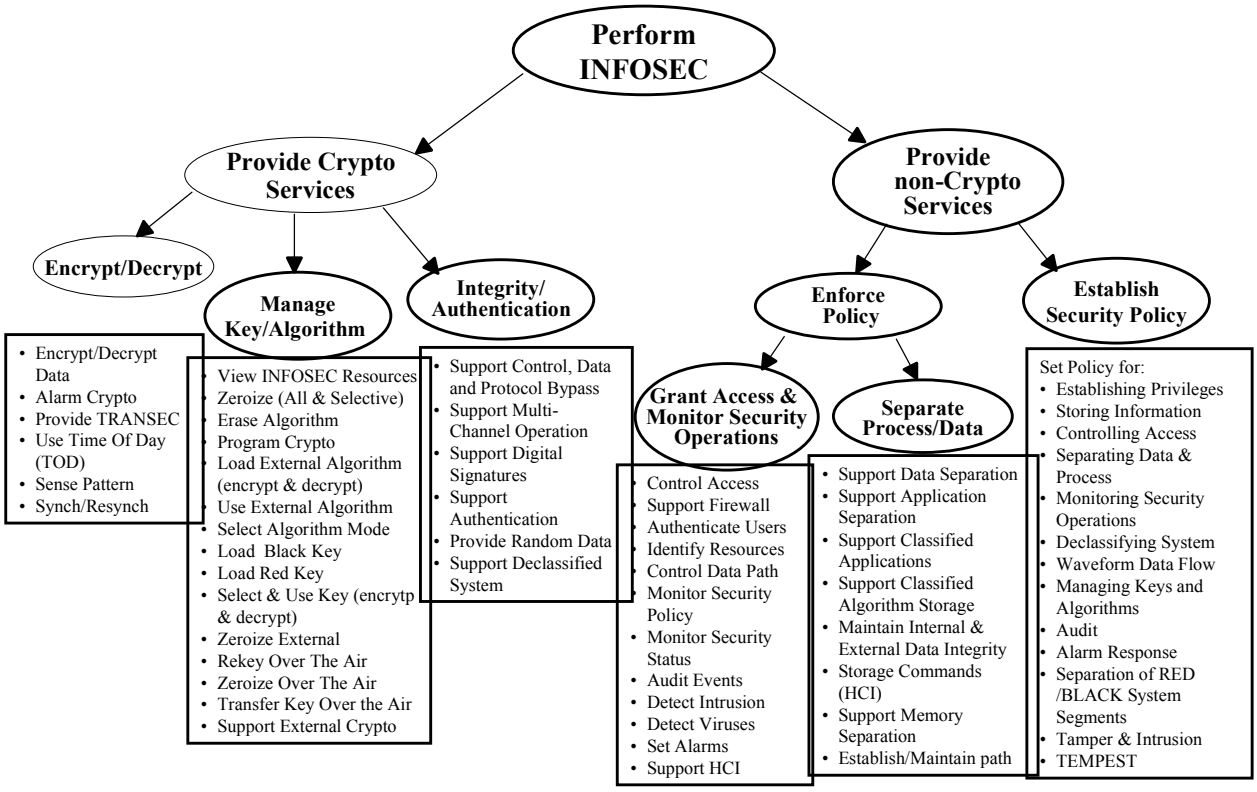

Figure 2-1. Security Functional Decomposition

Table 2-1 shows the allocation of each of these security functions to one or more of the elements that make up the SCA: Core Framework (CF), Middleware (CORBA), Operating System (POSIX), Hardware Class, or Applications Software. The specific allocation depends on the implementation of a JTRS and the capability provided in the its hardware and software. Because the CF is a general framework, it supports different options for meeting security requirements. Section 3.1 describes how security functional requirements are met by elements of the as specified in the SCA specification. No additional requirements are imposed on the CF as a result of this allocation. Figure 2-2 shows how examples of *SecurityDevices* and *SecurityResources* inherit from the basic Core Framework elements. It also illustrates how Security APIs (reference section 3.2.2.2 of the SCA specification) relate to *SecurityDevices* and *SecurityResources*.

Section 4.2.3.4 of the SCA specification shows examples of how the general INFOSEC hardware class supports subclasses providing hardware (and software) solutions to security requirements.

Table 2-1. Allocation of Security Functional Requirements to SCA Elements

		SCA Element				
Security Service Category	**Function**	**Core Framework Interfaces and Services**	**Middleware Transport (CORBA)**	**Operating System**	**Hardware (INFOSEC Device)**	**Application Software**
Encrypt / Decrypt	Encrypt Decrypt Data				x	
	Alarm Crypto				x	x
	Provide TRANSEC				x	x
	Use Time of Day (TOD)				x	x
	Sense Pattern					x
	Synchronize/Resynchronize				x	x
Manage Keys and Algorithms	View INFOSEC Resources	x			x	x
	Zeroize (All & Selective)				x	x
	Erase Algorithm	x		x	x	
	Load Internal Algorithm	x			x	x
	Load External Algorithm	x			x	x
	Select Algorithm Mode				x	x
	Load Black Key				x	x
	Load Red Key				x	x
	Select and Use Key				x	x
	Zeroize (Internal and External)				x	x
	Rekey Over-the-Air				x	x
	Zeroize Over-the-Air				x	x
	Transfer Key Over-the-Air				x	x
	Transfer Algorithm Over-the-Air				x	x
	Support External INFOSEC	x			x	
Support Integrity and Authentication	Support Control, Data, and Protocol Bypass	x	x		x	x
	Support Multi-channel Operation	x	x		x	
	Support Digital Signature		x		x	x
	Support Authentication				x	x
	Provide Random Data				x	x
	Support Declassified System	x		x	x	x

Table 2-1. Allocation of Security Functional Requirements to SCA Elements - Continued

Security Service Category	Function	SCA Element: Core Framework Interfaces and Services	Middleware Transport (CORBA)	Operating System	Hardware (INFOSEC Device)	Application Software
Grant Access & Monitor Security Operations	Control Access	x	x	x	x	x
	Support Firewall	x			x	x
	Authenticate Users			x	x	x
	Identify Resources	x	x		x	x
	Control Data Path	x	x		x	x
	Monitor Security Policy	x	x		x	x
	Monitor Security Status	x	x		x	x
	Audit Events	x	x	x	x	x
	Detect Intrusion				x	x
	Detect Viruses			x		
	Set Alarms				x	x
	Support HCI		x			x
Separate Processes & Data	Support Data Separation	x		x		
	Support Application Separation	x		x		
	Support Classified Applications	x			x	x
	Support Classified Algorithm Storage	x		x	x	
	Maintain Internal & External Data Integrity	x	x		x	x
	Storage Commands (HCI)	x				x
	Support Memory Separation	x		x	x	
	Establish & Maintain Data Paths	x	x			
Establish Security Policy Set Policy For:	Establishing Privileges				x	x
	Storing Information				x	x
	Controlling Access				x	x
	Separating Processes & Data				x	x
	Monitoring Security Operations				x	x
	Declassifying System				x	x
	Waveform Data Flow				x	x
	Managing Keys & Algorithms				x	x
	Auditing				x	x
	Alarm Responses				x	x
	Separation of Red & Black System Segments				x	x
	Tamper & Intrusion				x	x
	TEMPEST				x	x

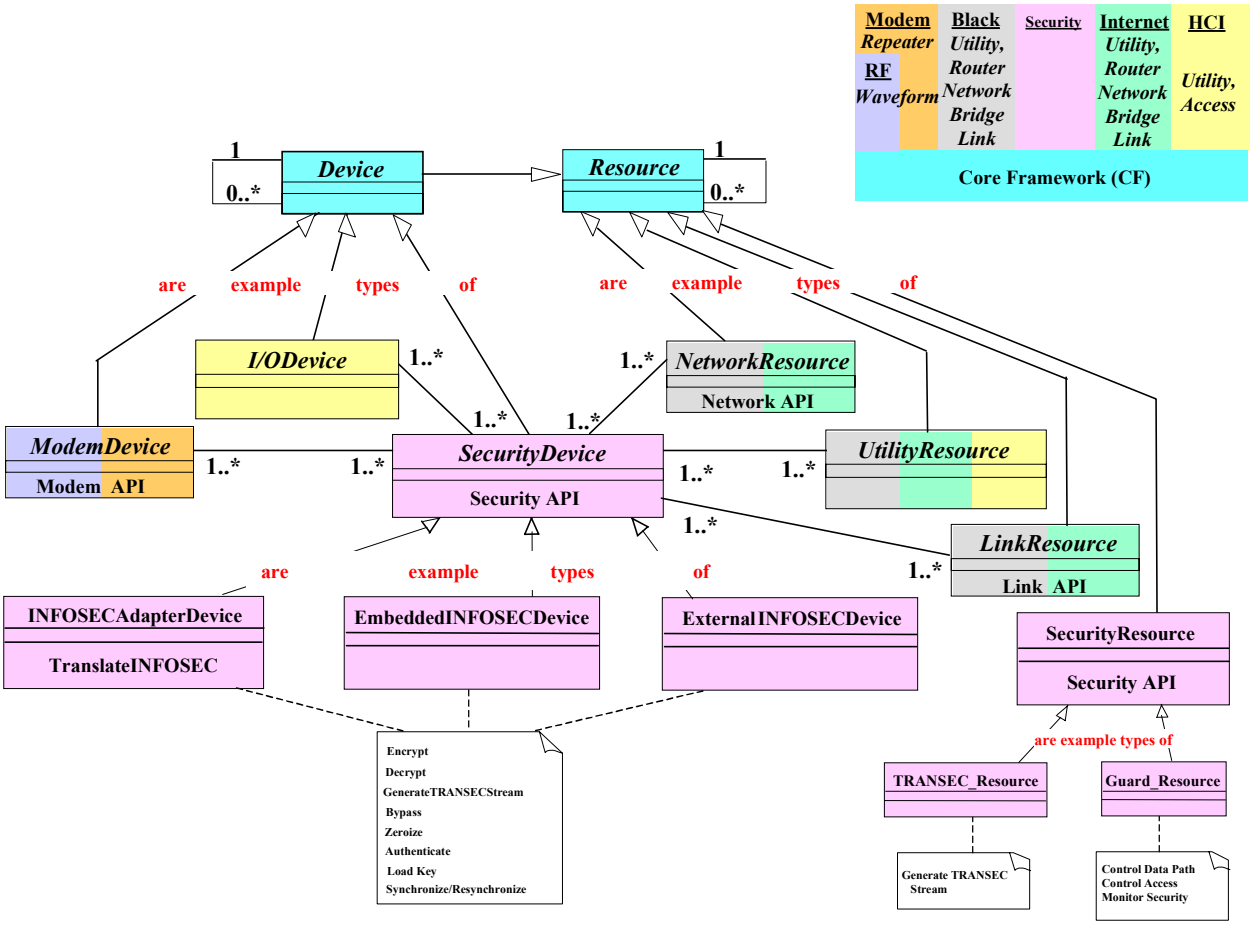

Figure 2-2. Examples of Security Devices and Resources

2.2 ARCHITECTURAL OVERVIEW.

The Security elements of the JTRS Architecture provide required Security Services in a multi-channel environment, where the channels may be at different Security Levels. The JTRS ORD requires System-High operation with eventual transition to MILS and MLS. System-High operation presents operational difficulties since it precludes concurrent operation of Plain Text and Cipher Text channels. Thus, different Security Levels should be considered from the beginning. This section describes elements of the architecture that support this approach.

The architecture provides for a Trusted Path between the information source and encryption *Resource*. The Trusted Path guarantees source authentication, channel separation and data integrity of the information-bearing data stream at some level of trust, specified by the Operational Security Doctrine (OSD) which is implemented through the Establish Security Policy function of Figure 2-1. The Trusted Path provides the required security services under the Integrity/Authentication and Enforce Policy requirements of that figure. The Trusted Path can be implemented by a variety of hardware devices and/or software components such as physical separation, services of Trusted Operating Systems, security services of CORBA, cryptographic signatures, and point-to-point encryption. An implementer selects the elements that are appropriate to the specific JTRS and implement them with a level of trust such that the layered

effect supports the trust required by the system's OSD. (It should be clearly understood that this discussion is architectural and conceptual in nature, rather than a specification of any one implementation.)

Traditional radio systems with embedded security have only one OSD that describes how the radio would be used for a particular application. This doctrine also describes any restrictions or specific operational procedures that have to be used. The JTRS requires an OSD that contains the specific user domain information including doctrine for each waveform application in the implementation. This OSD will need to contain the relevant security information for a particular waveform. An example of this information would be what cryptographic algorithm is to be used with the instantiated waveform. This composite OSD is the Security Policy for the JTRS implementation.

The creation and use of a trusted path can be realized by several means. One example is implementing physically separate hardware paths. A Trusted O/S could be used throughout the JTRS to provide the proper level of trust to separate traffic channels. Another example is to use a security guard to isolate traffic channel data. As such, one can abstract a security guard bus that permits only guards to release data to the guard bus.

Conceptually, the security guard has two functions. The first function is to uniquely manipulate data through a mathematical process. This process can change the data or add to the original data a label. The second function is to validate the guard bus data such that:

1) The data was not changed in any way.

2) The data that is trying to pass through the guard is valid for that guard.

During a waveform instantiation, producer and consumer *Ports* are created to move data. *Ports* will be uniquely created for each channel that is associated with each waveform. Security Guards can be created and assigned with one and only one channel. A consequence of the above guard definition is that Channel A's guard can not pass data through Channel B's guard.

Figure 2-1 illustrates a simple use case diagram for a guard that applies labels to the data.

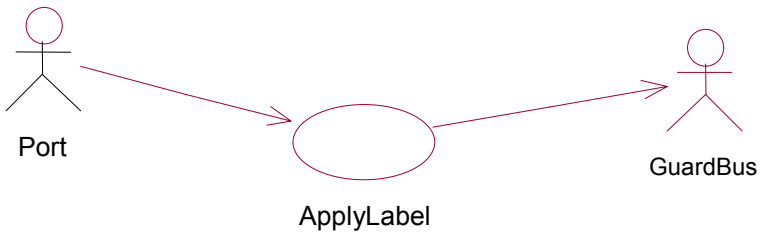

Figure 2-3. ApplyLabel Use Case

The diagram shows that once data has passed from a port to the ApplyLabel use case, data and its label appear on the guard bus that was discussed earlier. Once on the guard bus, only guards can validate labels and allow data to move through the system.

Figure 2-4 illustrates a second use case showing how the data is processed from the guard bus.

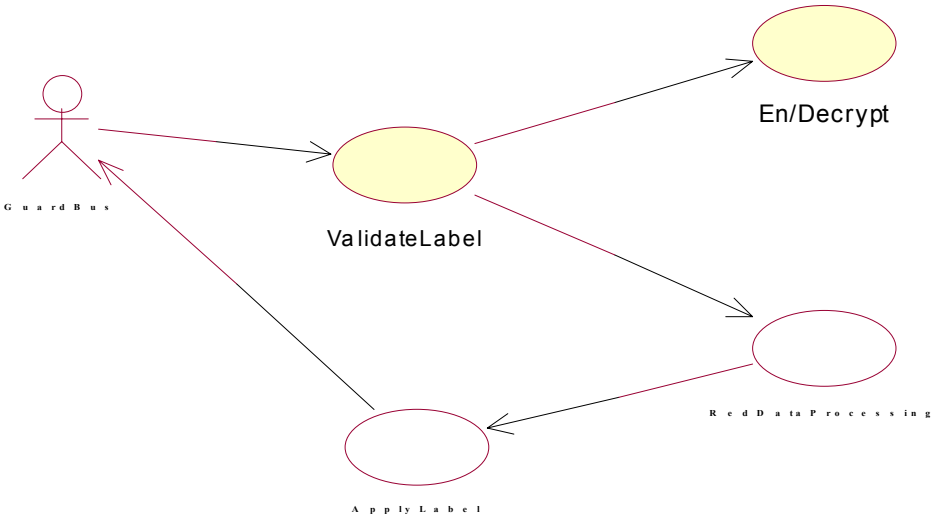

Figure 2-4. Processing of Data from the Guard Bus

The ValidateLabel use case checks that the accompanying label is correct, strips the label and releases the data for Encryption or RedDataProcessing use cases. For the RedDataProcessing use case, the label needs to be recalculated and applied before the data can move onto the GuardBus.

Multiple types of guards can be created. One implementation is with the creation of a *Port* for a waveform instantiation. The class diagram in Figure 2-5 depicts how this guard can be created.

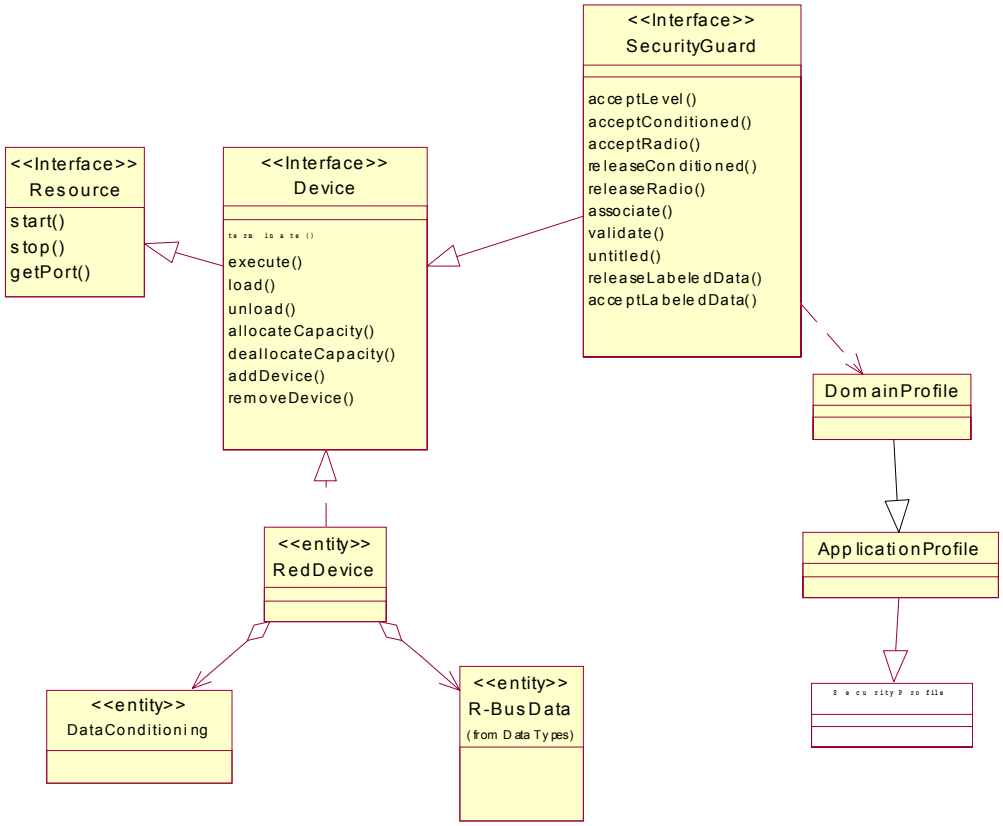

Figure 2-5. Example of Creation of a Port–Specific Guard

The second type of guard can be created with the security device, see Figure 2-6. This guard does not use the supplied configuration file. It independently builds the guard with the channel information.

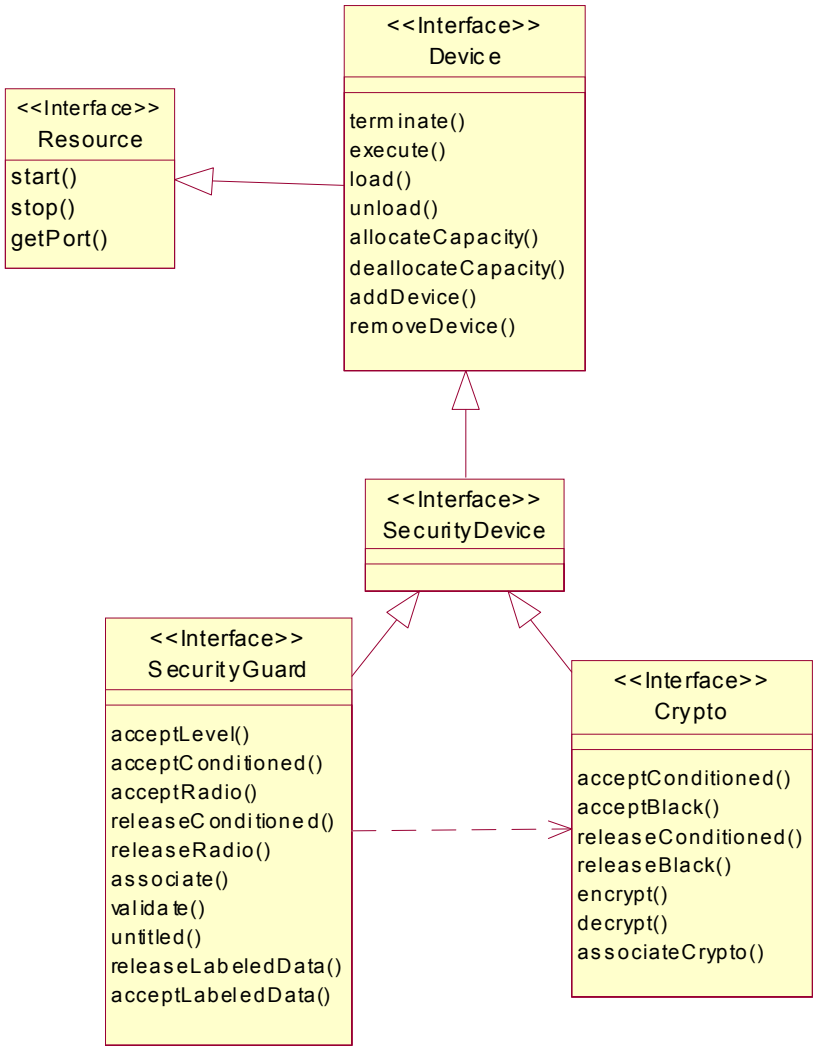

Figure 2-6. Example Security Device Guard Creation

3 SECURITY CLASSES AND SERVICES.

Each of the categories of Security Services identified in 2.1 is further broken down into user or operational interfaces and control or management interfaces. User interfaces describe functions associated with application users. Control interfaces are intended to describe functions associated with the control of the security services and will come from CF classes and applications or the classes of 2.2. [1]

3.1 CF SECURITY FUNCTIONS.

Table 2-1 allocates required security functions to elements of the SCA. For those allocated to the Core Framework, this section describes how they are met. The CF interfaces and behavior are extendible by applications software running security services.

3.1.1 View INFOSEC Resources.

The system configures Applications (virtual waveform channels) based on resource availability. Each JTRS implementation has a Domain Profile listing all system resources that are available to applications upon system initialization. For INFOSEC classes of devices, the attributes include encrypt/decrypt algorithms available and loaded, encrypt/decrypt devices available, and keys available and loaded (with their associated algorithm or device).

3.1.2 Erase Algorithm.

When an INFOSEC application using a programmable Encrypt/Decrypt algorithm, created by a CF *ApplicationFactory,* is terminated, the CF *Application releaseObject* operation tears down the application and causes the algorithm to be removed from the operational memory of the INFOSEC processor. This action does not remove the algorithm from any non-volatile storage element on the INFOSEC physical module. If the erasure is to be from non-volatile storage, that function will be provided by a security application that could be an instantiation of a *SecurityResource.*

3.1.3 Load Internal Algorithm.

Based on the Software Profile for a waveform application, the CF *ApplicationFactory* will load the appropriate cryptographic algorithm onto the INFOSEC module processor (or select the appropriate proxy for a non-programmable encryption device) when the application is instantiated.

3.1.4 Load External Algorithm.

Same process as in the previous section except the source of the algorithm is from an approved external source (analogous to a present day KeyFill device).

---

[1] Refer to the Open Group Website (http://www.opengroup.org/security) for additional information on a possible open standards approach to Security Services. This information is still preliminary and will be discussed with the JPO Security representatives and evaluated subsequent to the release of V1.0.

3.1.5 Support External INFOSEC.

The presence of an external INFOSEC element for use by applications will be part of an implementation's Domain Profile. A *DomainManager* can thus make use of this resource for waveform instantiation. *{This capability is not defined in the v1.0 Domain Profile.}*

3.1.6 Support Control, Data, and Protocol ByPass.

When a JTRS System is turned on or when a waveform application is instantiated, the specific security services required for ByPass will be implemented by the *ApplicationFactory*.

3.1.7 Support Multi-Channel Operation.

The Domain Profile for a JTRS implementation will contain the specific attributes, which define multi-channel capability for the SecurityDevices and SecurityResources of the system. If these are not present, multiple channels can not be instantiated by the *ApplicationFactory*.

3.1.8 Support Declassified System.

System Status reported via the CF *Logger* maintains the current status of all *Resources* within the system. The profiles of these *Resources* contain an attribute identifying each *Resource* as either unclassified or classified. The software application that shuts down the system can thus verify that no classified elements remain in system memory prior to complete shutdown.

3.1.9 Control Access.

The *SecurityGuard* is the means by which access is controlled within the system (data passage between connections) and to the system from outside. The specific method can vary widely on implementation. The resource that is available to the system will be used when Boot-up and each waveform are instantiated.

3.1.10 Support Firewall.

A means of controlling access from external elements is a Firewall. In JTRS, Firewall would be a *Resource* run as a separate application that is instantiated on Boot-up.

3.1.11 Authenticate Users.

When Access Control is instantiated upon Boot-up, the security services required for authentication will be used and implemented in that software. Because authentication information is sensitive, the information defining authentication methods and which uses have authorized access shall be stored in a protected space.

3.1.12 Identify Resources.

Applications loaded into a JTRS will be authenticated (e.g. with a digital signature) prior to being accepted.

3.1.13 Control Data Path.

In a multi-channel capable system, the *SecurityGuard* ensures proper access and connection. The *SecurityGuard* reports abnormal conditions through the CF *Logger*. \

3.1.14 Monitor Security Policy.

Security Policy for a JTRS implementation is the composite of Security Policy for the physical system and the waveform applications that are resident and can be implemented. The CF *Logger* will capture alarms representing deviations from a particular policy item. Security applications software provides appropriate responses for each alarm.

3.1.15 Monitor Security Status.

Similar to the previous paragraph, security items specifically identified by policy will be monitored and alarms reported through the use of status information collected in the CF *Logger* service. Application software, instantiating a *SecurityResource,* provides the appropriate responses.

3.1.16 Audit Events.

Security Policy will define which specific events, related to security, require audit. Applications software will collect and store required information using the CF *Logger, stringConsumer,* and *File* services.

3.1.17 Support Data Separation.

Resource information will contain attributes that identify an operating system's ability to support data separation. For applications that require this capability, software profiles will contain their dependencies for it. During a CF *ApplicationFactory create* operation, these dependencies will be compared to the properties of CF *Devices* to make sure that they can be deployed.

3.1.18 Support Process Separation.

Same as 3.1.17 except the attribute identifies the operating system with the ability to support separate processing spaces.

3.1.19 Support Classified Applications.

Classified applications refer to applications whose object code is classified. The SCA provides for applications code to be encrypted and decrypted through the INFOSEC class. An attribute, for the *SecurityResource*, identifies the classification level of the software application. The *Device*, on which the code will execute, is identified by its attributes as being on the Red side of the system. The device profile also defines whether the device is capable of executing in protected processing space. During setup of the classified software, the *SecurityGuard* insures that the software is only loaded into its proper device. Upon de-installation of a classified application, the CF *ApplicationFactory* insures that the code is erased from the device.

3.1.20 Support Classified Algorithm Storage.

Hardware devices authorized to store classified algorithms will be identified in the Domain Profile upon device registration (Boot-up). Classified algorithms, identified by their attributes, can only be loaded and stored on authorized devices.

3.1.21 Maintain Internal and External Data Integrity.

The CF supports *SecurityResources* that provide security services, such as MISSI for maintaining data integrity. These can be implemented as part of a *SecurityGuard* service or separately.

3.1.22 Storage Commands (HCI).

Applications are required to use the CF *File* services. Any HCI-based applications will therefore use the CF *File* services.

3.1.23 Support Memory Separation.

Same as 3.1.17 except that the required attribute identifies whether the operating system supports separate memory storage space.

3.1.24 Establish Data Paths.

The CF *ApplicationFactory* instantiates a waveform and, as part of that process, establishes the data paths that are connected together to implement the waveform.

3.2 SECURITY BYPASS.

3.2.1 Introduction.

Bypass, in a secure communication system, is the means by which information transfers from Red to Black without undergoing encryption or decryption. Since Bypass represents opportunities for inadvertent transfer of classified information from the Red to Black and ultimately over the air, stringent security requirements apply. For the SCA, Bypass is isolated within the Security Application (as viewed in the example in Figure 2-2 as a Guard_Resouce) and in the INFOSEC Class of Hardware in Section 4. Isolation of this important system capability to a particular physical element and its S/W proxies, within the SCA, simplifies definition and implementation. Figure 3-1 illustrates Bypass and the different types for Radio System operation and shows its relationship to the CORBA Middleware buses. Also shown is the “normal” path whereby the Red data passes through the encryptor to become Black and vice versa. Ideally, CORBA is the means of transferring messages between objects and the bypass function is transparent. In practice (implementation), the services required to implement bypass will impose some restrictions on CORBA use.

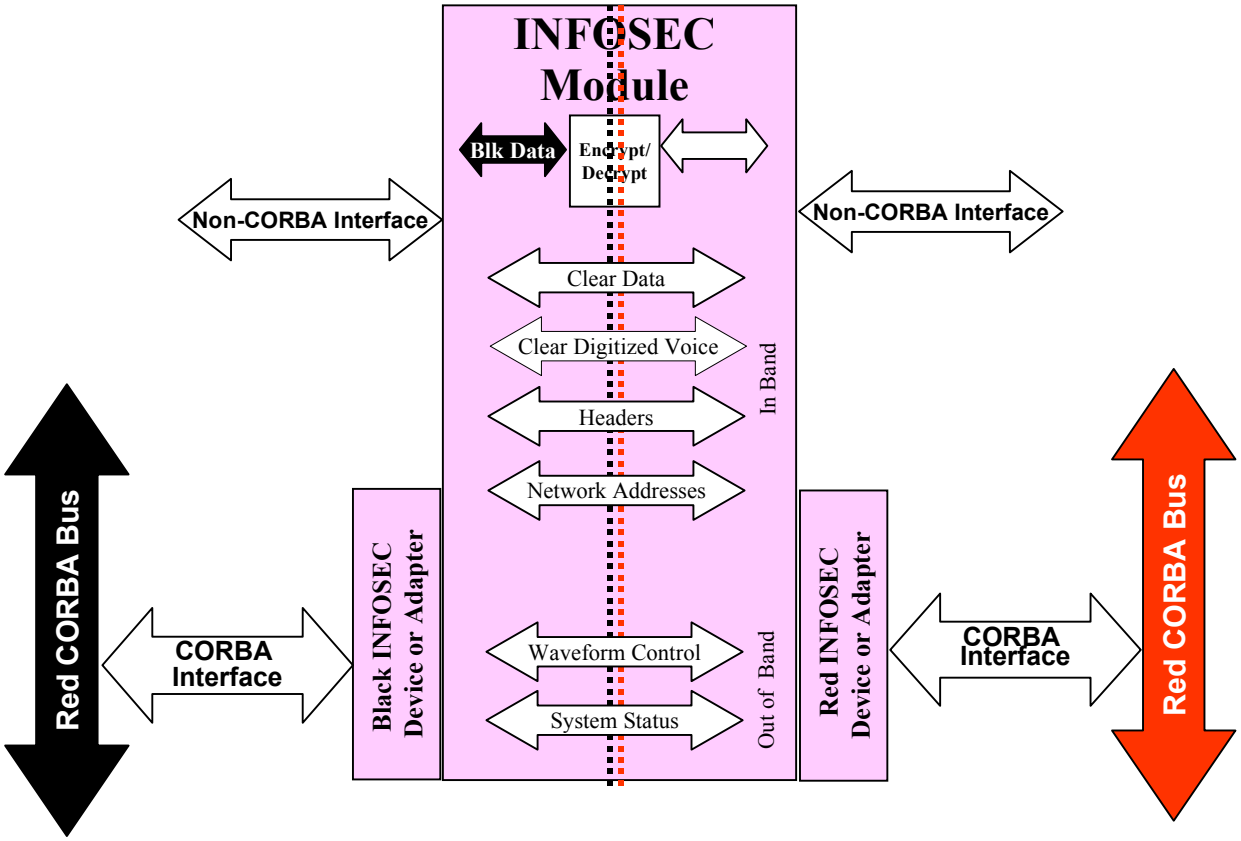

Figure 3-1. INFOSEC Bypass

3.2.2 Types of Bypass.

There are two categories of bypass that may require different levels of assurance because the risk of inadvertent disclosure of classified information is different. The first is In Band bypass where unclassified data, requiring bypass, is in the same data stream or same data path as the classified traffic that must pass through encryption (when the waveform mode of operation is secure). The second category is Out of Band bypass where a separate date path (logical or physical) is present. In general, for the first case the bypassed information must be associated with the data that is to be encrypted; and in the second case there is no association between the bypassed information and the message data. The In Band bypass consists of:

1. Clear Data – unclassified digital messages (generally using the same path on the Red side for when data must be encrypted)
2. Clear Digitized Voice – unclassified digitized voice
3. Network Headers – clear data that must be interpreted on the Black side of JTRS
4. Network Routing – similar to clear headers except that headers are added on the Black side based on routing information from the Red side.

Out of Band bypass consists of:

5. Waveform Control – information used by a specific waveform application for controlling systems resources during operation of the waveform
6. System Status – management information used by the system to configure itself, report events, report alarms; independent of any specific waveform

3.2.3 Services Required.

Access control services, provided by the *SecurityGuard* as well as service provided by the *SecurityDevice* (e.g., implemented in hardware on a physical module or associated with a mode of operation of an encryption algorithm or device), are the primary means for implementing bypass within the SCA. When implemented either in hardware or software or a combination of both, traditional methods like flow restriction and or message length can be used.

3.3 GENERAL SECURITY REQUIREMENTS.

3.3.1 Security and Alternate Networking Transfer Mechanisms.

If a networking waveform, using an alternate networking transfer mechanism (per SCAS section 3.2.2.2.2.3), is ported to a system designed for the standard networking transfer mechanism, the analysis provided shall verify how the alternative mechanism satisfies security services that are provided by the original system. This security analysis should address at a minimum: security services of the alternative transport and their use in meeting security requirements for the network data, how security bypass requirements are met by the alternative transport, and how protection from covert channel entry through shared internal and external OSI protocols is obtained. NSA provides the specific requirements for security analysis as part of the security endorsement program.

3.4 SECURITY APIs.

Security APIs shall be defined in a Service Definition, with a Transfer Mechanism, in accordance with 3.2.2.2 of the SCA specification and with the API Supplement to the SCA.

{Security APIs will be developed and will be included, possibly by reference.}

4 COMMON SERVICES AND DEPLOYMENT CONSIDERATIONS

4.1 COMMON SYSTEM SERVICES.

This section will define any common system security services that are not defined in the SCA but are required for JTRS implementations of the SCA. None have been identified at this time.

4.2 OPERATIONAL AND DEPLOYMENT CONSIDERATIONS.

This section will address common interfaces or features necessary to support deployment of JTRS SCA-compliant systems in the field. None have been identified at this time.

5 ARCHITECTURE COMPLIANCE

This section defines the criteria for certifying candidate system, hardware, and software application products to this supplement. Compliance to the SCA Specification is addressed in that document.

This supplement will be applied to procurements of multitude radio products and communication systems. In addition, this supplement may also be applied to hardware-only or software-only products that would be hosted on JTRS SCA-compliant systems.

5.1 CERTIFICATION AUTHORITY.

The JTRS Joint Program Office (JPO) holds the authority to certify that a candidate product meets the requirements of this supplement. This authority may be transferred, in time, to a general standards body.

5.2 RESPONSIBILITY FOR COMPLIANCE EVALUATION.

The responsibility for performing the evaluation of a candidate product's compliance is TBD. This body will determine the test methods and procedures used to establish compliance.

5.3 EVALUATING COMPLIANCE.

Compliance to this supplement is defined as meeting all requirements, except as specifically allowed herein. Products submitted as "SCA-Compliant for JTRS use" will be evaluated for compliance in accordance with the product validation methods and procedures established per section 5.2.